51CTO.com
技术成就梦想

我 们 只 谈 开 发

开发月刊

Development Monthly

2012年10月
总第019期



JavaScript打破所有规则

程序员的学习和积累

开发人员使数据库面临风险的十大方面

developer.51cto.com

目录

2012.09 Contents



编程排行 Billboard

3 10月编程语言排行榜：Dart发力，欲将JS赶下神坛

专题报道 《程序员和JavaScript》

5 JavaScript:打破所有规则

7 程序员的学习和积累

8 开发人员使数据库面临风险的十方面

技术热点 Techlogy hot

10 程序员小说 Out Of Memory

13 文科女生与IT

15 HTML5普及血泪史，外敌还是自绊

18 2013年十大急需的热门IT技能

19 那些年我体会过的袅袅余音

21 大数据存取选择：行存储还是列存储

23 MongoDB范围查询的索引优化

24 Visual Studio动态代码生成实现基础

25 Java的封面：关于程序员的嗅觉

27 浅谈大型网站的算法和架构

51CTO.com 技术成就梦想

出版方：北京无忧创想信息技术有限公司

责任编辑：彭凡　林师授　张伟　　封面设计：钱靓

联系方法：pengfan@51cto.com　010-68476606（分机 8058）

出版日期：2012 年 10 月 18 日

欢迎广大读者投稿

编者按

谷歌 Dart 在本月首次进入了 TIOBE 排名前 50 位。 Dart 是主要作为 Web 应用程序开发客户端的编程语言。更让人觉得异想天开的是，谷歌将 Dart 作为代替 JavaScript 的主要编程语言的 Web 开发。

10月编程语言排行榜：Dart发力，欲将JS赶下神坛

【51CTO 独家特稿】谷歌 Dart 在本月首次进入了 TIOBE 排名前 50 位，也是首次出色的表现。Dart 是主要作为开发客户端 Web 应用程序的编程语言。更让人觉得异想天开的是，谷歌将 Dart 作为代替 JavaScript 的主要编程语言的 Web 开发。微软，苹果和 Mozilla 目前已经宣布它们的浏览器将不支持 Dart 语言，所以更有趣的是，让我们关注 Google 如何在未来将 Dart 普及。

从上月份的编程语言排行榜来看，前 10 名编程语言排行没有太大的浮动。下面让我们来看看前 20 名的榜单。

2012 年 10 月编程语言排行榜榜单

Position Oct 2012	Position Oct 2011	Delta in Position	Programming Language	Ratings Oct 2012	Delta Oct 2011	Status
1	2	↑	C	19.822%	+2.11%	A
2	1	↓	Java	17.153%	-0.72%	A
3	6	↑↑↑	Objective-C	9.477%	+3.23%	A
4	3	↓	C++	9.260%	-0.19%	A
5	5	=	C#	6.530%	-0.19%	A
6	4	↓↓	PHP	5.669%	-1.15%	A
7	7	=	(Visual) Basic	5.120%	+0.57%	A
8	8	=	Python	3.895%	-0.01%	A
9	9	=	Perl	2.126%	-0.31%	A
10	11	↑	Ruby	1.802%	+0.28%	A
11	10	↓	JavaScript	1.201%	-0.93%	A
12	12	=	Delphi/Object Pascal	1.097%	-0.01%	A
13	13	=	Lisp	0.947%	-0.06%	A
14	18	↑↑↑↑	Pascal	0.863%	+0.12%	A
15	16	↑	Lua	0.728%	-0.07%	A
16	20	↑↑↑↑	Ada	0.651%	+0.04%	B
17	15	↓↓	PL/SQL	0.630%	0.27%	B
18	25	↑↑↑↑↑↑↑	Visual Basic .NET	0.589%	+0.12%	A-
19	21	↑↑	MATLAB	0.591%	+0.02%	B
20	19	↓	Assembly	0.568%	-0.05%	B

谷歌 Dart 编程语言

今年 10 月 10 日“Goto" 大会将在丹麦奥胡斯举行，谷歌两位开发人员会借着主题演讲的机会来揭晓这门新语言的全貌，直到那时我们才能了解 DART 的特点和功能。但幸运的是，“Goto” 会议网站为我们提供了一些关于这两位发明人的详细传记，他们两位分别是 SAP、Sun 公司计算机科学家及 Java 语言的共同发明者吉拉德 · 布拉查 (Gilad Bracha)，以及 Chrome 浏览器 V8 JavaScript 引擎的创造者拉尔斯 · 巴克 (Lars Bak)。

到目前为止，谷歌已经发布了过一种名叫“Go” 的新语言。所以我们可以认为 Dart 不会是另外一种类似于 C 的面向系统语言。从它的别名“结构化的 Web 编程” 可以看出，它很有可能是某种驻留在浏览器上的解释性语言。所以，它更类似于 JavaScript 或 Python，而不是像 Java 或其他的编译性语言。不过，布拉查和巴克曾经在过去深入广泛地使用过 Smalltalk 语言，解释性的 Smalltalkesque 语言也可以归类为是一种“结构化的 Web 编程”，这算是给了我们一个明显的提示。

然而，新的面向 Web 编程语言面临的问题就是，目前人们早已拥有无数种维护良好的可用■

原文未完，请参考

http://developer.51cto.com/art/201210/358937.htm



JavaScript 是一种能让你的网页更加生动活泼的程式语言，也是目前网页中设计中最容易学又最方便的语言。你可以利用 JavaScript 轻易的做出亲切的欢迎讯息、漂亮的数字钟、有广告效果的跑马灯及简易的选举，还可以显示浏览器停留的时间。让这些特殊效果提高网页的可观性。

JavaScript 是一种基于对象和事件驱动并具有安全性能的解释型脚本语言，在 Web 应用中得到了非常广泛的应用。 JavaScript 最初的确是受 Java 启发而开始设计的，因此语法上有很多类似之处，许多名称和命名规范也借自 Java。但是实际上，它与 Java 本质上是不同的。

JavaScript 为 HTML 设计师提供了一种编程工具 HTML 创作者往往都不是程序员，但是 JavaScript 却是一种只拥有极其简单的语法的脚本语言！ JavaSrcipt 是一种高级脚本语言，具有以下 3 个优点：

（1）JavaScript 采用在 HTML 文本中嵌入小程序段的方式，开发过程非常简单，并且提高了响应速度。

（2）JavaScript 可以直接对用户或者客户的输入做出响应，而不需要经过 Web 服务器的程序，减少了客户浏览器与服务器之间的通信量、提高了速度。

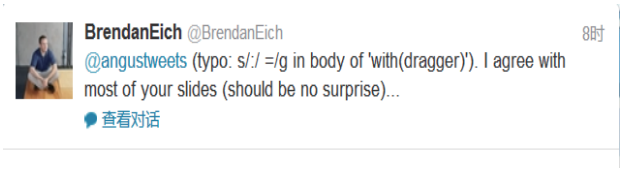
（3）JavaScript 是一种与平台无关的解释性脚本语言，依赖于浏览器，而与操作系统无关，只要计算机能运行浏览器，而且该浏览器支持 JavaScript，就可以执行 JavaScript 脚本程序。

51CTO 开发频道语

JavaScript:打破所有规则

最近，来自 Twitter 的前端工程师 Angus Crol, 在柏林举办的 JSConf 会议上，进行了题为 "Break all the Rulez" 的演讲，主要讲了一些我们通常认为是错误的不该使用的东西，其实是有用的。

北京时间今天凌晨，来自 Twitter 的前端工程师 Angus Crol, 在柏林举办的 JSConf 会议上，进行了题为 "Break all the Rulez" 的演讲，主要讲了一些我们通常认为是错误的不该使用的东西，其实是有用的 . 本文最下面有演讲用的 slides. 远在美国的 JavaScript 之父看了 slides 也说：我同意其中大部分观点 (看来还是有问题 ?).



下面我把要点简单翻译一下，不做扩展解释 .

with 语句

为什么不去使用它？

1. 意外的运行结果，可能隐式创建全局变量

2. 闭包作用域解析过多消耗

3. 后期编译

有人说 ,ES5 的严格模式可以防止隐式创建全局变量 (不用 var), 这样能减少 with 的一个问题 . 但是 ... 严格模式也不能使用 with 啊。

为什么说它是有用的？

1. 构建浏览器开发者工具

```
//Chrome Developer Tools
IS._evaluateOn =
  function(evalFunction, obj, expression) {
    IS._ensureCommandLineAPIInstalled();
    expression =
      "with (window._inspectorCommandLineAPI) {\
        with (window) { " + expression + " } }";
    return evalFunction.call(obj, expression);
}
```

2. 模拟块级作用域

```
//是的,还是这个老掉牙的问题
var addHandlers = function(nodes) {
  for (var i = 0; i < nodes.length; i++) {
    nodes[i].onclick =
      function(e) {alert(i);}
  }
};
```

```
//你可以通过在外面包一个函数来解决
var addHandlers = function(nodes) {
  for (var i = 0; i < nodes.length; i++) {
    nodes[i].onclick = function(i) {
      return function(e) {alert(i);};
    }(i);
  }
};
```

```
//或者使用'with'来模拟块级作用域
var addHandlers = function(nodes) {
  for (var i = 0; i < nodes.length; i++) {
    with ({i:i}) {
      nodes[i].onclick =
        function(e) {alert(i);}
    }
  }
};
```

eval 语句为什么不去使用它？

1. 代码注入

2. 无法进行闭包优化

3. 后期编译

为什么说它是有用的？

1. JSON.parse 不可用的时候

有人在 Stack Overflow 上说：

“JavaScript 的 eval 是不安全的，使用 json.org 上的 JSON 解析器来解析 JSON”

还有人说：“不要使用 eval 来解析 JSON! 用道格拉斯写的 json2.js!”

JavaScript: 打破所有规则 II

可是 :

```
// From JSON2.jsif (/^[\],:{}\s]*$/  .test(text.replace
(/*regEx*/, '@')  .replace(/*regEx*/, ']')  .replace
(/*regEx*/, ''))) {  j = eval('(' + text + ')');}
```

2. 浏览器的 JavaScript 控制台都是用 eval 实现的

在 Webkit 控制台或 JSBin 中执行下面代码

```
>(function () {    console.log(String(arguments.callee.caller))})()
function eval() {    [native code]}
```

John Resig 说过 :

“eval 和 with 是被轻视的 , 被误用的 , 被大部分 JavaScript 程序员公然谴责的 , 但如果能正确使用的话 , 可以用它们写出一些奇妙的 , 无法用其他功能实现的代码”

Function 构造函数

为什么说它是有用的 ?

1. 代码运行在可预见的作用域内

2. 只能动态创建全局变量

3. 不存在闭包优化的问题

用在了什么地方 ?

1. jQuery 的 parseJSON

```
// jQuery parseJSON

// Logic borrowed from http://json.org/json2.js
if (rvalidchars.test(data.replace(rvalidescape,"@")
  .replace( rvalidtokens,"]")
  .replace( rvalidbraces,""))) {

  return ( new Function( "return " + data ) )();
}
```

2.Underscore.js 的字符串内插

```

// If a variable is not specified,
// place data values in local scope.
if (!settings.variable)
  source = 'with(obj||{}){\n' + source + '}\n';

//..

var render = new Function(
  settings.variable || 'obj', '_', source);
```

== 运算符

为什么不去使用它 ?

1. 强制将两边的操作数转换为相同类型

为什么说它是有用的 ?

1. 强制将两边的操作数转换为相同类型

2.undefined == null

```
//这样写是不是很麻烦
if ((x === null) || (x === undefined))

//完全可以下面这样写
if (x == null)
```

3. 当两边的操作数类型明显相同时使用

```
typeof thing == "function";  //typeof运算符肯定返回字符串
myArray.length == 2;         //length属性肯定返回数字
myString.indexOf('x') == 0;  //indeOf方法肯定返回数字
```

真值不一定 ==true, 假值不一定 ==false

```
if ("potato") {
  "potato" == true;  //false
}
```

Array 构造函数

为什么不去使用它 ?

1.new Array() 也是 evil 的 ?JSLint 也推荐使用 [].

为什么说它是有用的 ?

其他

不要扩展原生的原型对象

(es 5 shims 都这么干) ■

原文未完,请参考

http://developer.51cto.com/art/201210/359675.htm

程序员的学习和积累

对于一个程序员来说，学习的重要性相信大家都能意识得到。如何才能有效地学习，高效地学习？探讨下这个问题，我认为很有必要。

每个人学习的方法和习惯可能都不尽相同，很多人习惯了在学校的学习方式，特别是很多刚出校门的人。在他们潜意识中，学习只是应付考试的一种手段，考完就一切 OK 了，全部还给老师了也无所谓，反正考完之后下次不会再考这么课程了（大学尤其如此）。长期的应试教育会让很多人以考试的心态来对待学习，我也是受害者之一，翻翻我硬盘中散落的各种语言编写的 "hello world"，各种半成品，它们就像一张张 60 分、80 分的试卷悄无声息地躺在某个昏暗的角落里。

职业生涯的学习一个最大的特点是具有持续性，你需要在某个知识或技能上花费你大部分精力。这时候，应付考试的学习方法已经不再有效了，而知识积累显得尤为重要。没有积累，所有的学习的成果都是一张 60 分的试卷。当你下次使用所学知识的时候，你手头有的就只有那张 60 分的试卷，你的学习将重新从 0 开始，由于时间关系或其他原因，你这方面的学习可能在你达到可以考 60 分的水平时又戛然而止，当你后面又需要使用时，又从 0 开始学习。如此循环往复，一定时期内，你的水平将永远停留在 0-60 分中间。

职业生涯学习的另一个特点是，你有机会去教。“The best method to learn is to teach”，你可以将你的所学通过各种形式的分享传授给他人。通过教的方式你不仅能够巩固你所学的知识，更能够推销自己、结交志同道合的朋友。

那么，基于职业生涯学习的特点，应该养成哪些好习惯呢，本人愚见，提出几点：

积累：

1、使用 note。比如 evernote，one note（个人推荐神器 evernote）。作为一名程序员，如果你不使用 note 的话，显然你已经 out 了。你可以使用 note 来在任意时间记录你的想法、学习笔记、分享、待办事项等等任意你想记录的。

2、经营好你自己的资产。你的作品、代码、idea、你搜集到的资料都是你的资产。对其进行分门别类，同步到网络硬盘（注意信息安全）、多备份都是很好的选择。

分享：

1、将自己的知识总结成 ppt、写出 demo。要养成这种意识，当你形成 ppt、demo 的过程中，你将站在更高的层次去看待你学过的知识、加深你的理解。当然，如果有机会一定要去讲，不管是何种级别的分享，哪怕是自己给自己讲或是讲给自己身边的朋友。

2、写技术博客。博客有很多种，对博客的定位，决定你写什么样的博文。如果你的博文每篇都作为一个研究课题来写，我相信你打造一个精品博客一定是早晚的事情。当然我们可能没有要求这么高，那么博客也可以当 note 来使用，这样可能有好心的人会指出你博文中的错误，和你探讨相关的技术等等。（本文未完，请阅读原文）■

http://developer.51cto.com/art/201209/356994.htm

开发人员使数据库面临风险的十方面

虽然这种对关键数据的简易访问已经极大地提高了工作人员的效率，并提高了顾客的购买欲，但它也为关键数据库打开了巨大的风险之门。不幸的是，许多风险是由缺乏资源的开发人员带来的，他们往往无法得到足够的时间、金钱、教育以及来自管理人员的支持，因而无法设计开发没有漏洞的应用程序。

在上述资源无法到位时，开发人员往往会犯这些错误：

1、过于相信输入方法

当今开发人员将数据库置于风险之中的一种重要方法是，使其应用程序遭受 SQL 注入攻击。当开发人员过于相信用户输入时，就往往会出现 SQL 注入漏洞。

如果不对其进行验证，例如，如果不验证一个只接受数字的电话号码字段，开发者就有可能允许黑客对数据库进行终端访问。例如，插入到表单域中的单引号有可能过早地关闭应用程序本应实施的合法 SQL 查询，从而使攻击者可以构建并提交新的查询。

解决这种问题的最好方法之一就是使用参数化查询。

参数化查询可以防止攻击者改变查询中的逻辑或代码，并阻止大多数 SQL 注入攻击。参数化查询出现的时间就像数据库一样长，它不像安全的其它许多领域，使用参数化查询也是保障性能和代码可维护性的最佳方法。

但开发人员不应当停留于此。还要对用户输入进行净化和验证，因为有时参数化并不可行。即使可行，攻击者也可以将非验证的输入用于其它恶意目的，如跨站脚本攻击等。

最老套且最糟糕的问题是，不对用户输入进行净化，即在对数据库运行查询之前，不剔除用户输入中并不需要的字符，例如不移除撇号或分号。

2、数据库错误消息显示给终端用户

在应用程序的 SQL 查询出现问题时，如果开发人员允许弹出特定的错误消息，这也许有助于诊断，但这也向攻击者提供了一个探查后端数据库的内部工作机制的很好途径。

通过得到的数据库错误消息，攻击者就会了解数据库的组织结构和应用程序查询等许多重要信息，从而更容易展开攻击。

错误消息可以泄露关于应用程序连接的数据库类型、底层设计等的线索。

这种错误会为盲目 SQL 注入攻击打开大门，所以开发者应当在页面显示一般性的错误消息而非特定消息。

3、轻率地对待口令

为了图方便，许多开发者用多种不安全的方法来轻率地对待用户口令。例如，他们可能将口令硬编码到应用程序中。

最明显的数据库风险之一在于硬编码口令和存在于配置文件中的口令。这两种设计选择都依赖于这种脆弱的安全性，他们假定攻击者不会访问这种信息，因而开发者并不关心文件或其中信息的安全性。

开发人员使数据库面临风险的十方面 II

同样危险的还有将口令存放到纯文本中的方法，其中包括不对用户输入的口令进行哈希，以及并不对哈希加盐（加盐：将随机位添加到哈希中）。

此外，还有不健全的口令管理，没有将强健的口令认证构建到应用程序中。

4、使所有的连接都是“超级”的

同样的，许多开发者通过“根”或其它一些超级用户账户来将应用程序连接到数据库中，从而将数据库置于风险之中。这通常是一个与硬编码口令的应用程序有关的问题，这是因为在这些情况下很难实施适当的特权管理。

普通的应用操作很少需要由“超级”特权所许可的访问，允许应用程序使用超级特权用户，会为不适当的非授权活动创造机会。所有的应用程序都应当使用最少特权的用户凭证连接到数据库。

5、相信存储过程是 SQL 注入的解决之道

当今的许多开发人员相信，存储过程是防止 SQL 注入的一种可靠方法。

事实上，如果存储过程自身的代码中包含漏洞，或者如果存储过程被以一种不安全的方式调用，它就并不能防止 SQL 注入。

在存储过程中可以发生串的连接或并置。即使存储过程的调用是准备好的语句，如果在存储过程自身内部构建和执行查询，它也有可能易遭受攻击。

6、将调试代码留放到生产环境中

正如厨师在做完了美味佳肴之后要打扫厨房一样，开发人员必须在将程序投放到生产环境中之前，必须清理其代码，以免打开数据库的后门。导致这种后门最常见的且易被遗忘的要素是遗留到生产环境中调试代码。

开发人员将后门放到程序中，其目的是为了调试，从而可以直接启用数据库的查询。忘记在生产环境中清除这种调试代码是愚蠢的，但也是很常见的错误。

7、劣质加密

比不使用加密更为糟糕的唯一问题是，错误地使用加密，因为这这种加密给企业一种虚假的安全感。

问题存在于细节中，不管是一个哈希函数或是一个加密例程。如果你并不知道如何正确地实施加密，那就将任务委托给一位专家，并且不要过早地将应用程序投放到生产环境中。

开发人员应当谨慎地对待其加密技术方面的技能和技巧。当今的黑客喜欢本地的加密设计，因为这种设计通常很差劲。

本地的加密几乎不能给有经验的攻击者提供什么价值，并且会给企业带来一种虚假的安全感。

8、盲目相信第三方代码

使用第三方的代码可能为开发人员节约大量的时间，但是开发人员不能在测试代码问题上抄近路，以确保所复制的代码不会给应用程序带来易受攻击的代码。

开发人员需要理解，自己要为整个应用程序的威胁分析负责，而不仅仅为自己编写的代码负责。

同样地，如果开发者希望限制那些给数据库带来风险的漏洞数量，他们就应当使用最新的开发框架。

许多开发人员会使用 WordPress、Joomla 或其它类型的应用程序框架，却没有用最新的安全补丁保持其最新。许多类似的电脑都易遭受攻击，应用程序也如此。

9、轻率地实施 REST（表述性状态转移）架构

本文未完，更多内容请查看原文■

http://developer.51cto.com/art/201210/359709.htm

程序员小说 Out Of Memory

我也是第一次知道我的世界竟然如此惨淡，不光被同样不多手多脚的人类歧视，甚至连这个冷冰冰藏在屏幕里都不敢和我相见的编译器也欺负我。

（一）

Terminal 里提示编译错误，原因是 World view。

看来我真的很久没有写程序了，不知道现在计算机语言进化的如此之快，甚至能够判断程序员的世界观。我也是第一次知道我的世界竟然如此惨淡，不光被同样不多手多脚的人类歧视，甚至连这个冷冰冰藏在屏幕里都不敢和我相见的编译器也欺负我。那到底我的世界观是什么？什么是我的世界观？

“别睡了！起来吃饭！”

我习惯性的擦擦留了一桌子的口水，这种无奈的习惯背后一定会有一个悲惨的人生。从小到大定然是受尽凌辱和嘲讽，想想在小学课堂上，老师吼道：“寇德！你给我醒醒！”当我还在梦里拉着临班小姑娘的手想进一步发展时，恍惚间这句话就让我拖着比红领巾还长的鼻涕口水甩了起来：“为什么要给你醒醒？我又没睡你。”

哦，对了，我叫寇德。小学三年级时，第一次开英语课，老师教了每个人几句介绍自己的话，到我这里的时候战战兢兢的站了起来，说道：“My name is KouDe.”

美丽而性感的女老师迟疑了几秒后说道：“你以后一定时一个做程序员的料子。”

当时我笑了。同学们都很不解，因为班上另外的六十二个学生里，三十一个想做国家领导人，三十一个想做大老板，但是他们不知道，其实我书包里一直偷偷藏着一本《未来之路》。珍贵到我甚至都不忍翻看，于是很多年后的今天我都不知道到底这本书里面说的是什么，光知道写这本书的人很有钱，然后他还是个写代码的。

不过我现在还是经常想到那个场景后独自感慨：“这人骂人真不带脏字。”但是老师也没有说错，就像买布料有好有坏一样，人亦是如此。

吃饭的时候看着卖相并不算好的菜和同样卖相的老婆，有些迟疑，就掏出手机给我一个不太熟的朋友发了条短信：“世界观到底是什么啊？”

在我刚刚准备重新拿起筷子去吃那只好像中了核辐射的鱼时短信提示就响了，一定是回复来了。一边老婆极其不悦的在给我使眼色，我知道她这个目光的暗语肯定不是说“我爱你”或者问我饭好不好吃，这个跟小时候老师不停的在劝说学生好好学习时一个意思，并不是真的说让你们好好学习，是防止假设因为你们没有好好学习所以混得不好，二十年后过来算账。当然把学习成绩很成就放在一起就好像是拿着甜甜圈沾咖啡一样的不靠谱，有个例外，除非社会畸形到拿收入去衡量人的价值。

面对着种种的压力，我还是犹犹豫豫的重新把手机拿了出来，短信内容是：“我他妈的哪知道，我是 Doctor of Philosophy，不代表我是学哲学的啊！”

程序员小说 Out Of Memory II

我做了极其强大的思想斗争后，决定还是要和他认真的讨论一下这个严肃的问题，回复道："哦。"

世界大约沉寂了十秒，短信来了："哦你***！你也是 Phd 吧！ Permanent Head Damage 吧！"

我无语了，我以为他至少会说我是"Paladin、Hunter、Death knight"。看来我当他是朋友，但是他还是不了解我的世界。

在我思路无限跳跃的时候，老婆已经收拾干净了桌子。现在如果不考虑别的，我至少能够确定我没有吃饱，但纵然饥肠辘辘我也不敢多问，知道自己错了那么默默接受惩罚好了。低头看看像怀胎五月的肚子，觉得腹中空空难耐，在那么一个瞬间我理解了朋友脾气不好的原因。

一年前他辞职跟朋友创业，但一直不顺，烧掉了存款又卖了房子，我问过他这个是何苦，他只是笑着告诉我，是为了梦想。

就这样奋斗了一年，前几天因为破显示器的分辨率过低无法支撑新的 iOS 系统的开发，所以朋友一狠心卖了早就空着的车位，给员工一人换了一台新的显示器。

显示器是在一个电商那里去订的，可能跟送货的沟通不好，就给送到楼下，死活不往上面搬了，再搬一件加二十。朋友的公司在十七层，七个人在一起合计了一下，决定自己往上抬，但保安又不让他们上电梯，说电梯是运人的，不是运货的。

公司一员工说："东西我们抱着成吗？"

保安在挖鼻屎："抱着也是货物。"

"那我的背包算不算货物啊？我衣服算不算货物啊？我假发算不算货物啊？我他妈的抱着老婆上去，老婆算不算货物啊！"

"你看，跑题了吧！"

朋友拦住了这两个人无聊的争吵："搬吧……"

就这样七个大男人抱着显示器就走上了救生通道。终于在九楼的时候，一个同事倒下了。之后在医院急诊室门口，跟无数遍重复的恶俗电视剧一样，出来一个医生，摇摇头。

我之后问朋友他的那台显示器怎么样了？

朋友沉默了半晌，用一个更加悲痛的表情看着我说："摔坏了。"

（二）

秋天是桂花味的。

早上起来在换灯泡的老婆问我，你们程序员换一个灯泡要多少个人，是不是要一个拧灯泡，一个扶椅子，一个在旁边指指点点还有一个人最后试试灯泡是不是拧好了。我放下了有点糊掉的炒饭，喝了一口两块钱的菊花茶，稍微品味了几秒钟，说道："这个是硬件问题，不归我们管。"

"你是跟我开玩笑呢？还是诚心找我乐呢？到底你是傻啊？还是笨啊？你是不是不打算跟我正常交流啊？"换好灯泡的老婆走到了我面前质问我。

可以看出，这个早上她心情很不好，从另外一个角度说，每个日复一日要赶一个小时路程去上班的白领早上心情都不会很好。这是一天绝望生活的开始。那种无助感就如同有人拿着枪顶着后脑勺，让你给自己挖一块敞亮的坟墓一样。

我又喝了口茶，这口是纯粹为了思考里面混乱的逻辑关系，然后坚定的回答道："是。"

老婆摔门出去，我心里却平静异常。

程序员小说 Out Of Memory Ⅲ

高中的时候班上有一个非常漂亮的女生，那个时候学校这个小江湖里流传了很多关于这个女生的谣言，有人说她家里很有钱，是个大家闺秀，是个小公主；有人说她其实条件一般，只是被个有钱人包养了；还有些比较过分的言论是说干脆这个女孩就是做鸡的，白天在学校报道，晚上去夜总会报道。当然，我知道这些都是谎话，都是谣传，江湖不就是这个样子吗，我说大侠，我就是大侠，我说他是傻逼，他就是傻逼，每个人心里都有一块自己的腥风血雨。

其实这个女生就和我住邻居，我们从小玩到大。平时回家都在一起学习，但从来不一起上下学，她也邀请过我一起走，每每这个时候我都会告诉她，我比较喜欢另外一条路上的煎饼，那个味道实在太好了。她犹犹豫豫的问我，能不能带她去吃，我说不行，这是我自己的宝藏。

就这样，我们在从同一个起点出发，通往同一个终点的路上竟然走了两条不同的道路。

在高二那年，她决定要出国留学，高三下半学期已经拿到了录取通知就干脆没有上。那时因为语言成绩很好，平时各方面也很优秀，加上一些名头很响亮的竞赛奖就硬生生的拿了一个斯坦福的录取信，这个消息在我们学校就是平地一声雷。于是流言又传开了，有人说她被学校招生的潜规则了；有人说她花了大价钱给学校捐钱买的录取；有人说她根本没有拿到通知，在骗人。总之就这样，她又在风口浪尖坚持了几个星期后销声匿迹。

在我们高考前她回来过一次，拿了一本同学录让我们都签一下。全班只有我没写，她问我为什么，我说再过两天这些东西网上就有了。她有些不悦，我安慰到："没事，反正以后还会再见。"

没想到我居然说中了，或者说，在约定面前，茫茫人海间隔的不过是一根网线。于是，我们又要再见了。在十年以后。

我重新联系上她是在 Facebook 上找到的。不对，是被找到的。她突然加我好友，问我是不是那个寇德。

我们聊的很开心，我看她是在 Facebook 工作就问她那工作环境如何。她说还可以，但是现在已经辞职了。

"辞职？"

"对啊，准备去创业了。对了，我记得高中的时候你就在做 indie 游戏，后面好像也听说你自己办公司，做的如何啊？"

我转眼看看龟缩在破烂居民楼里的哥几个，说道："还成。"

"风投进来第几轮了？"

"还没有……"

"哦，那就是盈利模式很清晰了，开始就能盈利真的很难得啊！好厉害哦～"

"你们呢？"

"我们几个 co-founder 凑了 50 万，然后又拿了 50 万的天使，现在东西刚刚做出来，正在找投资，你那有钱呗？给我们投点？"

"暂时没有……"

"那好吧～哈～那……我过段回国，一起出来见见吃个饭？"

（本文未完，请阅读原文）■

http://developer.51cto.com/art/201210/358939.htm

文科女生与IT

就这样，我把这次经历当作一场闹剧，付出的代价是一个月的时间和辛劳，没有报酬还倒贴了不少交通费，收获的是独自闯荡社会的勇气和看人做事更加成熟稳重。

第一次在这里写东西，想记录下在工作中的苦与乐，还有一些感想。

毕业至今大概 1 年零 4 个月，刚换到第二份工作快满两个月了。现在想想，是应该经常经常总结才行。

说来好笑，原本我的人生应该和 IT 这个行业毫无关联，现在却真真实实地每天打着代码，戴着所谓程序员的帽子。

我爱好广泛，特别是文艺方面，又比较能说，小时候还做过当主持人的梦。稍微大了一点，也还是想走文艺工作者的路，可发现自己不长个儿了。高考前还特地打电话问艺考的老师，说是除非特别优秀，并且拿过权威性的奖，那不到 1 米 6 是没有关系的。好吧，只能说那时不懂得争取，就凭这么一句话就浇灭了儿时的梦想。

之后稀里糊涂地念了日语专业。毕业前正想按部就班地找和日语有关工作时，正好一家对日外包企业到学校招人。想着反正和日语有关，对软件什么的也比较感兴趣，而且自认为在女生当中算是对 IT 方面有些天赋，于是从毕业前 3 个多月就进入公司开始了艰难的女程序员之路。

上班之前曾非常慌张，还上网发帖问大家该怎么办。有人鼓励有人讽刺，不管怎样，还是硬着头皮忐忑忑忑地去上班了。

我想说的是，感谢所有回复的网友，鼓励的让我更有勇气，讽刺的让我提前感受到个别极端自以为是的前辈嘲讽新人的感觉，以至于能让我在后来的工作当中心态平和了许多。

后来经常在网上看到如“女生适合在 IT 里做什么”或者“女生适不适合编程”之类的问题。毕竟我做的时间比较短，说不上经验、建议什么的，只是自己的想法而已。我觉得性别不重要，兴趣是第一，努力是第二，天赋是第三，个人觉得缺一不可。其实各行各业都是一样的，一般人都知道这个道理。可惜也有不少人没认识到这一点，摸爬滚打了许多年还是在当初的职位干着当初的工作，没什么起色，还经常怨天尤人。当然非常热爱某一项工作愿意干一辈子、并且能够做到极致的那些人另当别论，他们是值得钦佩的。

网上有人说，拼技术肯定拼不过男程序员，劝女生做测试、DBA、需求、UI 等等。我觉得，一般情况下，这些确实比写代码要轻松一点点。可有时候也特别辛苦。所以我劝女生们，要是想进入 IT 这个行业，不管做什么，都要有不怕吃苦、不怕学习的觉悟。你要是能吃苦，又能不断学习提升自己，能够独当一面后，必定在 IT 行业里的职位可以随你挑，除非你的上司不懂珍惜人才，或者本身人品有问题，装作看不见你这块金子，那劝你赶紧辞职。

说说我的第一份工作吧，做的是软件外包。

文科女生与 IT II

先说好的地方。公司比较正规，规章制度都比较完善，进公司时的新人培训（不是技术和业务培训）要一个礼拜。培训内容有企业文化介绍、公司各项制度介绍、职业素养、个人职业发展规划、体能拓展，培训讲师有公司内部的，也有外聘的。

个人觉得这个新人培训做得比较成功。至少在工作当中几乎没有同事跑到行政人事部门问有关制度的事情。各种制度的文档也很全，归类也比较好，放在公司服务器上，查阅也非常方便。至于制度定得合不合理，倒是说不出个一二，反正大公司的制度都差不多。

第二个好的地方是 5S 做的很好，环境不错。有专门的质量保证部门。所以很少有项目会出什么严重的问题。

第三点挺矛盾的，是好的地方也是不好的地方。安全保密做得太好，普通员工不能上网，主机上的 USB 插口也被锁了，甚至有的开发室不能带包包和手机。对老板来说这太好了，员工上班时间的利用率很高。对员工来说，太痛苦了，除了工作还是工作。部门领导比较开明的话还比较好，看看技术方面的书也行，有的领导看大家比较空，还会叫人开班让大家学东西。

想了几分钟，终于挤出第 4 点。稳定，那是相当地稳定，几乎不会开除员工。待遇福利不太好，导致流动性非常大。不用公司开除，人走得也是很快了。所以只要你有大专学历及以上，正常智商，品性不坏，就算零基础，也不会炒了你。待遇说低么也比公务员稍微好一些，所以比较适合那些稳稳当当过小日子的人。

接下来就不细细罗列老东家的缺点了，只说一下我为什么辞职。很悲剧，被随机分到了最惨的一个部门，后来大家都把它称作“养老院”。部门经理是内部提拔的一个老程序员，技术上没问题，但是的确不是做管理的料子。他自己可能没有意识到这点或者意识到了也不愿意改变。

个人觉得做部门经理应该不断主动发现内部问题，尽量合理安排属下解决问题，必要时指点一二。可我们这位老大每天埋头研究项目的技术难点。其他事情只求平安无事。下属有时提出来说产品有哪里哪里可以改进，老大会说，只要客户没提出来咱们就不管。就是这种无所作为的思想和态度，逼走了很多同事。

做得少当然错的少。搞笑的是，上面的领导只看到了“错的少”，每年都还会通报表扬我们老大。然后我们老大的老婆（也在我们部门）非常骄傲地跟我们说“我们部门都是精英啊”。无语。

当部门离职率排到公司第一名的时候，领导终于找老大谈话了。失望的是，老大说“公司待遇不好留不住人”，不知道是“官官相护”，还是那个领导真蠢，总之，就是不了了之。这也是我彻底心灰意冷决定离开的主要原因。

我不怕加班也不讨厌加班，但确实在原来的公司加班很严重，所以辞职之后，在家好好休息了一段时间，也想了很多。

怀疑自己一年前的决定是否错误，是不是回归本行，做做翻译什么的或许更适合我。但总觉得不喜欢，对于不喜欢的工作，更不愿意去尝试了。（本文未完，请阅读原文）■

http://developer.51cto.com/art/201209/358035.htm

HTML5普及血泪史，外敌还是自绊

然而事实是，每当我用电脑打开国内的主流网站，Flash 仍然在狠　我的脸让我在 HTML5 的春秋大梦中醒过来，铺满整个屏幕的依旧是煎鸡蛋专业户 Flash 而非 HTML5。这不得不让人生疑，HTML5 在业界风风火火了这么多年，莫非到头来只是一个空口号？

近几年来，HTML5 的大潮似乎是势不可挡。在桌面应用环境中，各大浏览器陆续卷进了 Chrome 引发的跑分竞赛，对 HTML5 的支持力度如同磕了药般疯狂；在移动平台上，Flash 先后在 iOS、Windows Phone 和 Android 上吃瘪，Android 把 Flash 逐出家门后 HTML5 已经成为了移动互联网的唯一选择。放眼望去，仿佛四野皆充满着 HTML5 千秋万代一统江湖的高呼。

HTML5 声势浩大

然而事实是，每当我用电脑打开国内的主流网站，Flash 仍然在狠打我的脸让我在 HTML5 的春秋大梦中醒过来，铺满整个屏幕的依旧是煎鸡蛋专业户 Flash 而非 HTML5。这不得不让人生疑，HTML5 在业界风风火火了这么多年，莫非到头来只是一个空口号？ Flash 到底还会在互联网上纵横多久，再让 CPU 煎熟多少个鸡蛋？很明显，HTML5 的推广，并没有如同大多数人想象般那样一帆风顺。

那么到底是什么因素在阻拦着 HTML5 的普及呢？让我们一起来看看。

拦路虎：Flash

说起 Flash，不少人对它的印象都是又蠢又慢，还经常会把浏览器崩溃得七零八落。不过你可能想不到的是，在 1996 年 Flash 诞生伊始，它可是互联网上冉冉升起的超级新星——小巧、高效、跨平台，无论是开发者还是用户，看到 Flash 都会笑逐颜开。



Flash 诞生之初是互联网世界的超级英雄

随着年岁的增长，在 Adobe 丰盈的奶水的滋润下，Flash 茁壮成长。Flash 先后加入了对 Javascrip、HTML、XML 的支持，新增了滤镜和层的模式，加持了 Actionscript 语言，并增强了影音方面的功能一举把 RM/RMVB 赶到了线下。Flash 甚至被人誉为是“最灵活的平台”，易于开发且移植性强的特点让它在互联网上打出了一片天地。

然而，Flash 的成长似乎茁壮得有点过了头。Flash 的功能越来越强大，开发人员使用 Flash 开发的产品也愈发繁多庞杂，Flash 开始被用来干一些奇怪的事情，比如说卖卖小广告啦，玩玩小游戏啦——这些应用多是粗制滥造，很难想象有人会为 Flash 广告优化代码，是吧。人们慢慢发现，当低劣的 Flash 充斥着网页的时候，电脑开始不堪重负了，Flash 落下了一个烤肉机驱动程序的称号。

外敌重重自绊自脚？ HTML5 普及的血泪史 Ⅱ

Adobe 把 Flash 养得太肥了

但是实际上，Flash 的效率并不低，只是 Flash 的滥用和参差不齐的 Web 前端开发人员造成了一系列不良后果。毕竟 Flash 功能强大而且易上手，什么人都会用 Flash 进行开发，产出了各种奇葩东西也在情理之中，结果就是这些胡编乱造的 Flash 让人们的电脑假死崩溃了一次又一次。

Flash 的效率低下主要归咎于开发者，但其安全性和稳定性欠佳就和 Adobe 不上心有关系了。君不见多少个 0day 漏洞由 Flash 而出，多少次要浏览器主动为更好的 Flash 稳定性而更新。安全性和稳定性，直到现在依然是 Flash 的软肋。



Flash 的崩溃让多少人抓狂

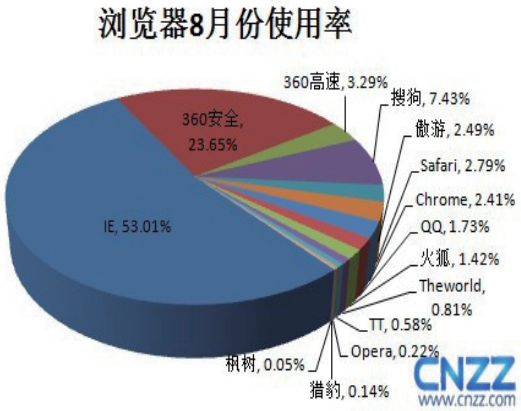
总的来说，Flash 的优点和缺点都很明显，但是 Flash 的缺点无碍于它成为网络上的霸主——并不是每个人都会关心什么病毒不病毒，浏览器崩溃了重新打开也能恢复标签页。和 HTML5 相比，Flash 功能不输，且易于开发，对浏览器的依赖十分小，人们只需要升级 Flash 插件就能享受到最新版本。相反，如果你没有装备一个先进的浏览器，HTML5 根本不会出现在你面前。Flash，这个 Adobe 和开发者的宠儿，现在的 HTML5 仍未能撬动它的根基。

绊脚石：浏览器

前面提到，Flash 能够制霸网络的很大一个原因就是对浏览器的依赖小，Adobe 早已经在 Flash 中加入静默升级机制，用户不知不觉中即可享受到最新版本 Flash 带来的特性，这一切都无需依赖浏览器。

HTML5 则不然，如果 HTML5 想要部署于广众大罗的电脑上，这就得先把这些为数巨大的电脑通通安装上能辨认 HTML5 的浏览器，只有当人们不再使用能把 CSS3 谱写的哆啦 A 梦渲染成方框怪兽的浏览器时，HTML5 才会展现它应有的魅力。然而，旧习惯的惰性力量惊人，这在中国又尤为明显。IE6 从 21 世纪初发布开始，十年如一日，至今仍在统治着人们浏览互联网的窗口，新型浏览器在中国举步维艰。下面是今年 8 月份中国浏览器的排行榜单，IE 的强势可见一斑。

今年 8 月份中国市场浏览器使用率情况



在榜单中，IE 仍然以过半的使用率独占鳌头——这个还是保守情况，榜单中可没有统计使用了 IE 内核的“第三方浏览器”。

外敌重重自绊自脚？ HTML5 普及的血泪史 Ⅲ

由于中国的特殊国情，大部分用户在电脑买回来后就基本不会去动那些 Ghost XP、Ghost Win7 之流的系统，这些系统内置的 IE6、IE8 自然也成为了人们上网的标配——很遗憾，无论是 IE6 还是 IE8，对 HTML5 的支持度都极差，HTML5 眼看 Flash 在这些老旧的浏览器中风生水起，自己却力所不及，只能 45 度仰望天空为的是不让眼泪掉下来。

所幸这一切只发生在桌面应用环境中，在移动平台，HTML5 已经成为了新时代互联网的象征。在这里，HTML5 可得大大地感谢苹果。苹果在 2007 年发布 iPhone 时，并未提供对 Flash 的支持，当时苹果的官方说法是，硬件效能不足以支撑 Flash 运行。当时 Adobe 还心存一丝侥幸，以为随着硬件的发展，iOS 支持 Flash 是迟早的事情。于是 Adobe 在 Flash CS5 上开始支持将 Flash 输出成 iOS App，但始终无法在 Safari 上使用 Flash。

苹果将 Flash 逐出了 iOS

Adobe 和苹果都倍受设计师的青睐，双方之间也有过不少合作，Adobe 万万没有想过苹果在移动平台上对 Flash 居然会如此强硬。面对 Adobe 在 iOS 上运行 Flash 的满怀希冀的眼神，苹果不理不睬，转身去和 W3C、Mozilla 和 Opera 合作，大力推广 HTML5。得益于苹果在移动平台上的强势，HTML5 在移动互联网遍地开花。而 Windows Phone 和 Android 看到 iOS 和 HTML5 的合作竟隐隐有一统江湖之势，于是纷纷表示 Flash 什么的弱爆了。在 Android 4.1 抛弃了尾大不掉的 Flash 后，HTML5 成为了移动平台唯一的选择，也是面对 Flash 唯一取胜的战场。

Android 4.1 放弃 Flash，HTML5 得以制霸移动市场

移动平台 HTML5 得以战胜 Flash，主要是因为苹果等厂商对 Flash 的强硬态度，在移动平台上用户也没有太多的选择，厂商说什么就是什么——你总不能把 iOS 刷成 Android 吧。桌面平台用户的自由度就大多了，微软可不敢把你的 IE6 强卸掉后强推 IE9。HTML5 想要全面取代 Flash，任重道远呐。



目作孽不可活：HTML5 本身

虽然前文我们一直在说 HTML5，但这实际上是一个泛指。前文所说的 HTML5 包括了 HTML5、CSS3、JavaScript 以及新的浏览器 API 所包含的整个 Web 前端技术，这一套技术并没有一个严格的统一的标准——HTML5 仍是一个草案，而各家浏览器的 API 的内容又不一样，可以说 HTML5 本身就十分零散。

HTML5 拥有众多出彩的特性，可以让人们在网上直接绘图作画、听歌看片、吹水聊天以及将 Web 应用转化称本地程序等等（虽然这些 Flash 并非做不到，但是很少人用 Flash 用心做这些），HTML5 成为了最新移动技术的代名词。HTML5 甚至可以构建一个移动操作系统——Firefox OS 中就使用了 HTML5 编写系统应用。（本文未完，请阅读原文）■

http://developer.51cto.com/art/201209/357036.htm

2013年十大急需的热门IT技能

高管们也表示他们更需要的是能将现有技术转变成能在商界中竞争的有力产品。下面就是该调查结果总结的 2013 年急需的十大热门技能。

著名 IT 杂志《计算机世界》日前针对各大公司计划招聘 IT 专家而发起的一项预期调查显示，在 334 名 IT 界高管中，有 33% 的人称将在接下来的 12 个月中计划增设更多的 IT 业领头职位。当然，高管们也表示他们更需要的是能将现有技术转变成能在商界中竞争的有力产品。下面就是该调查结果总结的 2013 年急需的十大热门技能。

1、程序和应用开发技能

调查显示，有 60% 的 IT 高管们计划将在接下来的 12 个月招聘这样的人才。据资深人士里德（Reed）称：“对公司来说，技术和软件是提高生产力、降低成本以及创建更好网页形象的最佳利器。”他还补充说，IT 公司将需要能创造并提高这方面技术的员工。

2、项目管理技能

调查显示，该技能的需求获得了 40% 高管们的认可。据美国放贷公司 Quicken Loans 软件工程副总裁杰米 – 汉米尔顿（Jamie Hamilton）称，对项目管理职位的需求部分是因为应用与应用之间越来越紧密的联系增强了现有项目的复杂性。

3、咨询台 / 技术支持技能

据纽约蒙特法沃医疗中心（montefiore medical center）的副主席杰克 – 沃夫（Jack Wolf）称，他正寻求不仅会建立和使用系统而且还会给予其他员工技术支持的新员工，他说：“新的系统意味着你必须有更多的咨询台来处理更多的咨询量。”

4、安全技能

调查中，有 27% 名高管表示他们需要这项技能。安全一直是 IT 界高官们所关心的问题，为保护其系统以及越来越复杂的数据库，他们对安全专家的需求与日俱增。

5、商业智能 / 逻辑分析技能

该调查中，有 26% 的人认为这项技能很重要。最佳的候选人必须具备技术知识，商业知识以及强大的数据和数学分析背景。这项技能并不普通（本文未完，请阅读原文）■

http://developer.51cto.com/art/201209/357788.htm



下载通道：http://os.51cto.com/art/201209/358908.htm

《Linux 运维趋势》是由 51CTO 系统频道策划、针对 Linux/Unix 系统运维人员制作的一份开放的电子杂志，内容从基础的技巧心得、实际操作案例到中、高端的运维技术趋势等等。■

那些年我体会过的袅袅余音

曾否体会过，那年某月某日某人不经意的一句话，会让你茅塞顿开、眼前一亮？即使过了许多年，到了如今的某月某日你不经意想起，依然记忆犹新、意味深长？

虽然这些在人生中相遇的“某人”并不是伟人，“言语”也不是什么励志名言，但在特定环境特定时间特定关系特定人物的基础上，一句看似普通的话会产生化学作用，以至于像余音袅袅那样生生不息。因此我特意把带来印象深刻体会的“言语”摘录下来，希望能引起一些共鸣。

为了利于日后内容扩充，这些“言语”我尽量按时间轴排序。

『你给我最大的感觉是很特别』

初中临近毕业时，某位女生对我说过的一句话。

一直到现在，我都没有找到比“特别”更好的能归纳我性格的形容词。

『你是我大学期间少数值得我敬佩的同学之一』

大学毕业后几年，一位大学同学在跟我聊天后 QQ 里给我的留言。

我当时感到很惊讶，因为很难想象“这句话”会是出自这位个性张扬并且带点自傲的同学。其实我很想对他说：你也是值得我敬佩的朋友之一。

『收件人一栏里领导的姓名顺序要按职级排名』

第一家公司的坐我旁边的同事。

当时我第一次发邮件给 N 个领导（包括总部的几位领导），收件人一栏的领导排名没有在意排序，把副总的名字排在正总前面。旁边的女同事就提醒我“收件人一栏里领导的姓名顺序要按职级排名”，并告诉我每个领导的职级高低。

你所忽视细节可能是别人所关注的，虽然即便做对了 99 次也未必有人赞赏，但忽视一次可能足于让很多人感到失望。这是我新入职场时第一个大收获。

『你知道我为什么聘用你吗？因为你听懂了我所说的话』

第一家公司的部门副经理。

在我入职后的一次两人谈话中，经理闲聊中提到了上面这句话。

一个人能力太低或太高也不足以让上司感到满意，假设双方的想法总是无法找到交叉点甚至是背道而驰的话，不管是上司还是下属，最终都是不利于发展。“听懂”并不是意味着“听话”。如果是“听话”，那就是 machine；“听懂”才是管理者所期望的。

『这是应聘者该具备的技能，还需要准备什么？』

第一家公司的副经理。

当时公司在招人，我的朋友经我推荐进入笔试，经理觉得我的朋友试题做的不是很好，我当时处于对朋友的“庇护”，随便说了一句可能我的朋友没准备好。经理感到很诧异：“这是应聘者该具备的技能，还需要准备什么？”。在公司看来，

那些年我体会过的袅袅余音 II

老板付薪水是招聘合格员工是来为公司创造价值，而不是来学习的。往往新入职场的人很容易忽略这一点，在不熟悉的领域往往会爆出这样的论调“这个问题 / 技术不是我不懂，只是暂时忘了，如果给我一点时间了解这方面资料我很快可以解决 / 学会的”。

意外的是，经理最终的决定还是给了我的朋友一次机会。

『你的能力有多大，你的舞台就有多大』

现公司的一位部门经理，已离职。

当时我顺利通过公司的二面，经理最后问我还需要了解什么，我问“我在贵公司将来会有哪些发展空间？”

他很快回答说“你的能力有多大，你的舞台就有多大”，从他爽快的反应看来，估计他早准备好 100 个以上的答案来回应这些问题，显然他清楚这个回答会比较吸引我。



让我明白，去了解身边的人追求什么，这个很重要。

『你问我如何管理下属？跟朋友相处一样，就是这么简单』

第一家公司的部门经理。

那时候我已经离开了这家公司，偶然算是老朋友老同事相聚，我们会一起吃饭和了解双方的近况。我当时在新公司初涉项目管理，向眼前这位已经有近 20 年经验的朋友抛出了“如何管理下属”的话题。

“你问我如何管理下属？”他重复了我的话，大致是因为要思考一个不落俗套的回答。停顿了一下他说“跟朋友相处一样，就是这么简单”，最后还补充了一句“就像你我一样”。或许，“与下属成为朋友”并不是每个地方都适合，但也许这就是一种境界。

『我觉得项目经理至少要做到两点：1. 争取资源；2. 要扛得住』

现公司的部门经理。

当时某个项目出现了一些问题，正好该项目经理提出调部门申请，部门经理只好把他项目交由给其他人接管，正好我有参与过项目前期的规划和设计评审，自然而然这个任务就落在我的身上。

那次部门经理、前项目经理、我三人闭门谈话，部门经理明显感到很失望和气愤，对着前项目经理吼出了一句“我觉得项目经理至少要做到两点：1. 争取资源；2. 要扛得住”。

这句话我觉得正好概括了作为国内大部分企业的项目管理的“国情”，企业希望 PM 们应具备的基本素质——能扛得起“鸭梨”。

编者按

目前大数据存储有两种方案可供选择：行存储和列存储。业界对两种存储方案有很多争持，集中焦点是：谁能够更有效地处理海量数据，且兼顾安全、可靠、完整性。

大数据存取选择：行存储还是列存储

从目前发展情况看，关系数据库已经不适应这种巨大的存储量和计算要求，基本是淘汰出局。在已知的几种大数据处理软件中，Hadoop的HBase采用列存储，MongoDB是文档型的行存储，Lexst是二进制型的行存储。在这里，我不讨论这些软件的技术和优缺点，只围绕机械磁盘的物理特质，分析行存储和列存储的存储特点，以及由此产生的一些问题和解决办法。

一．结构布局

行存储数据排列

	Column 1	Column 2	Column 3	Column 4	Column 5
Row 1	Data 1-1	Data 1-2	Data 1-3	Data 1-4	Data 1-5
Row 2	Data 2-1	Data 2-2	Data 2-3	Data 2-4	Data 2-5
Row 3	Data 3-1	Data 3-2	Data 3-3	Data 3-4	Data 3-5

列存储数据排列

	Row 1	Row 2	Row 3
Column 1	Data 1-1	Data 2-1	Data 3-1
Column 2	Data 1-2	Data 2-2	Data 3-2
Column 3	Data 1-3	Data 2-3	Data 3-3
Column 4	Data 1-4	Data 2-4	Data 3-4
Column 5	Data 1-5	Data 2-5	Data 3-5

表格的灰色背景部分表示行列结构，白色背景部分表示数据的物理分布，两种存储的数据都是从上至下，从左向右的排列。行是列的组合，行存储以一行记录为单位，列存储以列数据集合单位，或称列族(column family)。行存储的读写过程是一致的，都是从第一列开始，到最后一列结束。列存储的读取是列数据集中的一段或者全部数据，写入时，一行记录被拆分为多列，每一列数据追加到对应列的末尾处。

二．对比。从上面表格可以看出，行存储的写入是一次完成。如果这种写入建立在操作系统的文件系统上，可以保证写入过程的成功或者失败，数据的完整性因此可以确定。列存储由于需要把一行记录拆分成单列保存，写入次数明显比行存储多，再加上磁头需要在盘片上移动和定位花费的时间，实际时间消耗会更大。所以，行存储在写入上占有很大的优势。

还有数据修改，这实际也是一次写入过程。不同的是，数据修改是对磁盘上的记录做删除标记。行存储是在指定位置写入一次，列存储是将磁盘定位到多个列上分别写入，这个过程仍是行存储的列数倍。所以，数据修改也是以行存储占优。 数据读取时，行存储通常将一行数据完全读出，如果只需要其中几列数据的情况，就会存在冗余列，出于缩短处理时间的考量，消除冗余列的过程通常是在内存中进行的。列存储每次读取的数据是集合的一段或者全部，如果读取多列时，就需要移动磁头，再次定位到下一列的位置继续读取。再谈两种存储的数据分布。由于列存储的每一列数据类型是同质的，不存在二义性问题。比如说某列数据类型为整型(int)，那么它的数据集合一定是整型数据。这种情况使数据解析变得十分容易。相比之下，行存储则要复杂得多，因为在一行记录中保存了多种类型的数据，数据解析需要在多种数据类型之间频繁转换，这个操作很消耗CPU，增加了解析的时间。所以，列存储的解析。

过程更有利于分析大数据

三．优化

显而易见，两种存储格式都有各自的优缺点：行存储的写入是一次性完成，消耗的时间比列存储少，并且能够保证数据的完整性，缺点是数据读取过程中会产生冗余数据，如果只有少量数据，此影响可以忽略；数量大可能会影响到数据的处理效率。列存储在写入效率、保证数据完整性上都不如行存储，它的优势是在读取过程，不会产生冗余数据，这对数据完整性要求不高的大数据处理领域，比如互联网，犹为重要。

改进集中在两方面：行存储读取过程中避免产生冗余数据，列存储提高读写效率。

如何改进它们的缺点，并保证优点呢？

行存储的改进：

减少冗余数据首先是用户在定义数据时避免冗余列的产生；其次是优化数据存储记录结构，保证从磁盘读出的数据进入内存后，能够被快速分解，消除冗余列。要知道，目前市场上即使最低端 CPU 和内存的速度也比机械磁盘快上 100-1000 倍。如果用上高端的硬件配置，这个处理过程还要更快。

列存储的两点改进：

1. 在计算机上安装多块硬盘，以多线程并行的方式读写它们。多块硬盘并行工作可以减少磁盘读写竞用，这种方式对提高处理效率优势十分明显。缺点是需要更多的硬盘，这会增加投入成本，在大规模数据处理应用中是不小的数目，运营商需要认真考虑这个问题。

2. 对写过程中的数据完整性问题，可考虑在写入过程中加入类似关系数据库的“回滚”机制，当某一列发生写入失败时，此前写入的数据全部失效，同时加入散列码校验，进一步保证数据完整性。

这两种存储方案还有一个共同改进的地方：频繁的小量的数据写入对磁盘影响很大，更好的解决办法是将数据在内存中暂时保存并整理，达到一定数量后，一次性写入磁盘，这样消耗时间更少一些。目前机械磁盘的写入速度在 20M-50M/ 秒之间，能够以批量的方式写入磁盘，效果也是不错的。

四．总结

两种存储格式各自的特性都决定了它们不可能是完美的解决方案。

如果首要考虑是数据的完整性和可靠性，那么行存储是不二选择，列存储只有在增加磁盘并改进软件设计后才能接近这样的目标。

如果以保存数据为主，行存储的写入性能比列存储高很多。

在需要频繁读取单列集合数据的应用中，列存储是最合适的。

如果每次读取多列，两个方案可酌情选择：采用行存储时，设计中应考虑减少或避免冗余列；若采用列存储方案，为保证读写入效率，每列数据尽可能分别保存到不同的磁盘上，多个线程并行读写各自的数据，这样避免了磁盘竞用的同时也提高了处理效率。

无论选择哪种方案，将同内容数据聚凑在一起都是必须的，这是减少磁头在磁盘上的移动，提高数据读取时间的有效办法。■

MongoDB范围查询的索引优化

我们知道，MongoDB 的索引是 B-Tree 结构的，和 MySQL 的索引非常类似。所以你应该听过这样的建议：创建索引的时候要考虑到 sort 操作，尽量把 sort 操作要用到的字段放到你的索引后面。但是有的情况下，这样做反而会使你的查询性能更低。

比如我们进行下面这样的查询：

```
db.collection.find({"country": "A"}).sort({"carsOwned": 1})
```

查询条件是{“country”：“A”}，按 carsOwned 字段的正序排序。所以索引就很好建了，直接建立 country，carsOwned 两个字段的联合索引即可。像这样：

```
db.collection.ensureIndex({"country": 1, "carsOwned": 1})
```

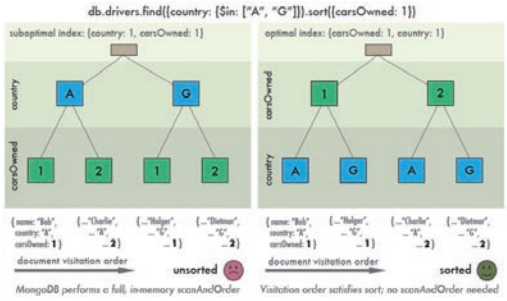
我们来看一个稍微复杂一点的查询：

```
db.collection.find({"country": {"$in": ["A", "G"]}}).sort({"carsOwned": 1})
```

这回我们是要查询 country 为 A 或者 G 的数据条目，结果同样按 carsOwned 字段排序。

如果我们还使用上面的索引，并且使用 explain() 分析一下这个查询，就会发现在输出中有一个“scanAndOrder” : true 的字段，并且 nscanned 的值可能会比想象中的大很多，甚至指定了 limit 也没什么效果。

这是什么原因呢，我们先看下面这张图：



如上图所未，左边一个是按 {“country”：1, “carsOwned”：1} 的顺序建立的索引。而右边是按 {“carsOwned”：1, ” country”：1} 顺序建立的索引。

如果我们执行上面的查询，通过左边的索引，我们需要将 country 值为 A 的（左图的左边一支）所有子节点以及 country 值为 G 的（左图的右边一支）所有子节点都取也来。然后再对取出来的这些数据按 carsOwned 值进行一次排序操作。

所以说上面 explain 输出了一个“scanAndOrder” : true 的提示，就是说这次查询，是先进行了 scan 获取到数据，再进行了独立的排序操作的。

那如果我们使用右边的索引来做查询，结果就不太一样了。我们没有将排序字段放在最后，而是放在了前面，相反把筛选字段放在了后面。那这样的结果就是：

我们会从值为 1 的节点开始遍历（右图的左边一支），当发现有 country 值为 A 或 G 的，就直接放到结果集中。当完成指定数量（指定 limit 个数）的查找后。我们就可以直接将结果返回了，因为这时候，所有的结果本身就是按 carsOwned 正序排列的。■

本文未完，更多内容请查看原文

http://database.51cto.com/art/201209/357491.htm

Visual Studio动态代码生成实现基础

本文作者之前想谈一谈如何通过引入框架或代码生成来解决（在某种程度上）项目开发实际会遇到的编码的问题。



这篇文章讨论以下 3 个问题：

1. 代码生成器应该做什么
2. 大多数代码生成器的缺点
3. 动态代码生成实现的基础

代码生成器应该做什么？

我认为，目标是加快项目开发，方式是减少重复代码手工操作，实现是用过代码生成技术。反过来说，就是代码生成要尽量让能自动化的代码不手动来操作。当然产生了很多附属的优点，如稳定性、便于测试、可以集中精力在业务逻辑上等，可是不能本末倒置。套用一句话，一切不以自动化为目的代码生成器都是耍流氓。

大多数代码生成器的缺点

现在大多数的（应该不是所有）代码生成器有一个最大的问题，就是多次生成导致的拷贝粘贴（无法动态响应类结构或表结构的变化）。现在的代码生成器大多数是在表结构和类代码之间单向生成，但不代表代码生成不能生成其他代码。就算只从生成持久化代码的角度考虑，这些代码生成器除了能根据某个时间点的状态生成一次性的代码，就没有什么价值了。而这些生成的代码如果用于实际项目（不可能不调整结构），又会进行多次生成，产生多次拷贝粘贴。这样的代码生成器其实就是 YY，只是 YY。当然 YY 有理（我就曾是其中一员），YY 无罪。我只想问问有几个人在项目中实际使用了？有几个人的项目连表结构和字段都没改过？ 我跟大家一样都曾经对代码生成器很感兴趣，都自己去 DIY 一个。我甚至在实际项目中尝试使用，可惜效果并不好。

理想的代码生成器

理想代码生成器在我看来应该有以下优点：

1. 能够集成到开发环境中
2. 能够随时根据模型或数据表的变化重新生成代码
3. 通过分部类或继承完全隔离手工代码出现在生成代码的文件中
4. 不要实体类和数据表之间直接映射

动态代码生成实现的基础

如果 Visual Studio 提供了对 2 种方式（从代码或数据表开始）代码生成的支持（如上所述，我不建议直接映射）：

1.CodeModelEvents 事件提供了对动态检测模型变化并重新生成数据表的支持。

2. 通过 AddIn 方式、T4 方式对数据库表进行动态检测并重新生成模型文件的支持。

在 AddIn 中也可以直接使用 CodeModel 或 FileCodleModel 直接进行代码生成，如果通过 T4，可以通过调用 Solution.FindProjectItem 找到 T4 文件，通过 Open 和 Save 方法让 T4 模板自动运行更新代码来达到动态更新的目的。■

本文未完，完整部分请看原文：

http://developer.51cto.com/art/201209/357352.htm

编者按

这篇短文起源于我和几个程序员曾经的一次关于为什么 Java 不可靠的谈话，这不是批评 Java，这只是一次关于程序员的嗅觉的学习案例。长时间以来，程序员们发展出了对好（或者坏的）技术的灵敏嗅觉。我想，把对我觉得 Java 不靠谱的原因写下来会是件非常有意思的事情。

Java的封面：关于程序员的嗅觉

关于译者

感谢 BOOM 提供了本文翻译。BOOM 是个程序员，就职于某 IT 公司，有关他的更多信息请他的新浪微博：http://weibo.com/huji0624

译者注：这篇文章是 Paul Graham 写于 2001 年的一篇预言 Java 的文章。

这篇短文起源于我和几个程序员曾经的一次关于为什么 Java 不可靠的谈话，这不是批评 Java，这只是一次关于程序员的嗅觉的学习案例。

长时间以来，程序员们发展出了对好（或者坏的）技术的灵敏嗅觉。我想，把对我觉得 Java 不靠谱的原因写下来会是件非常有意思的事情。

有些读过这篇文章的人觉得像这种方式提出一种新观点是一种有趣的尝试 . 其他人觉得似乎对自己不太了解的事情妄加评论会招来麻烦。所以，以防万一，我先澄清一下，我所写的这些不是针对 Java 的（我从来没有用过 Java），而是关于程序员的嗅觉的东西（关于这方面我想过很多）。

有句老话叫“你无法根据封面看出一本书内容 (you can‘t tell a book by its cover)”，它源自于书被出售的时候，是以买主的个人喜好来装订封面的时代。在那个年代，你不可能通过书的封面辨别出它是哪本书。但随后出版商们也在进步，现代的出版商们让你很容易就通过书的封面来分辨一本书。

我在书店花费了大量的时间，我觉得到目前为止，我已经学会了理解出版商想告诉我的关于一本书的所有事情，或许更多。在余下我没有待在书店的时间里，我大部分时间都在电脑前，我觉得从某种程度上来说，我也学会了通过一种技术的外在面貌来评判它。有可能是运气的原因，我让自己免于了几种最终被证明很不靠谱的技术。

目前为止，我不太喜欢 Java。我从来没有写过哪怕一行 Java 代码，顶多瞟过一眼 Java 相关的参考书籍，但我有种预感，Java 不会成为一门很成功的语言。也许最后会被证明我搞错了；做技术预言是一件很危险的事。但从某意义上来说，把为什么我不喜欢 Java 的原因，作为一种时间胶囊保存起来，是值得的，以下就是这些原因：

1. 它被吹得有点太天花乱坠了。真正的标准是不需要推广的。没有人会去推广 C，Unix，或是 Html。一个真正的标准往往是随着时间的推移被建立起来的，在这个过程中大多数人已经听说过它了。在程序员的雷达屏幕上，Perl 阵营和 Java 差不多庞大，甚至还要大些，但它是依靠自己的长处逐步发展起来的。

2.Java 的目标太低。在最初的 Java 白皮书中，Gosling 明确表示，Java 被设计得让程序员更容易使用，而不像 C 语言那样。它被设计成了另一个 C++：C 加上一些从几个更高级的语言里拿过来

Java 的封面：关于程序员的嗅觉 II

的思想。就像情景喜剧和垃圾食品或是旅游团的发明者一样，Java 的设计者有意识地设计了一个供没有他们聪明的人使用的产品。从历史上来说，被设计成提供给他人使用的语言都不怎么样： Cobol, PL/I, Pascal, Ada, C++ 都是这样的。好的语言是那些设计者为自己创造的语言，比如 C, Perl, Smalltalk, Lisp.

3.Java 动机不纯。有人说过，如果人们只因为自己有些想说的话而写书，而不是为了写书而写书，这个世界将会更加美好。同理，我们总是对 Java 如雷贯耳不是因为它想表达一些关于编程语言的看法。而是因为 Sun 把它作为对付微软的计划之一。

4. 没有人真正喜欢它。C, Perl, Python, Smalltalk, 和 Lisp 序员都非常喜欢他们的编程语言。我从来没有听人说过他喜欢 Java。

5. 人们是被强制使用 Java 的。我认识的很多使用 Java 的人用 Java 写程序因为他们觉得他们不得不用 Java。要么是因为他们觉得他们必须用 Java 才能吸引到投资，要么是他们觉得顾客需要 Java，或者是他们的经理让他们使用 Java。他们都是很聪明的人；如果一门技术很好，那么人们应该是自愿使用它的。

6.Java 有太多维护人员。最好的编程语言一直以来都是被一小拨人开发出来的，但 Java 似乎是被一个委员会所维护的。如果 Java 最终被证明是一门好语言，那历史上会首次出现委员会设计出了一门好语言这样的事情。

7，官僚作风太重。根据我所知甚少的 Java 知识，似乎使用 Java 的时候有很多的规定需要遵从。真正优秀的编程语言不会是这样的。它会让你做你自己想做的事，然后摸索出一套自己的办法。

8. 太虚伪。Sun 目前假装 Java 是一门一如 Perl, Python 一样的开源草根语言，而它只是碰巧被大公司控制。所以，Java 也拥有了大公司的沉闷。

9.Java 是为一些大型组织所设计的，大型组织和程序员们有着不同的目标。他们希望一种适合于一大帮平庸的码农所使用的语言，这种语言的特点就是把愚蠢的人所犯的错误的破坏力减到最小，就像 U-Haul 卡车上的限速器一样。程序员们不喜欢居高临下的编程语言。程序员们只需要力量。历史上，为大型组织所设计的语言（PL/I,Ada）在竞争中败下阵来，程序员们的语言（C，Perl）最终赢得了胜利。所以有这么个说法：现在的青少年程序员，就是未来的 CTO。

10. 不靠谱的人才喜欢 Java。基本上我欣赏的大多数程序员都不迷 Java。那谁喜欢 Java？那些穿着西装的上班族，所了解的编程语言很少，但是他们经常从各种媒体上听说 Java；还有大公司的程序员，因为他们发现尽然有比 C++ 还好的东西；再有就是愣头青的准毕业生，会喜欢任何能给他带来一份工作的东西（他们比较在乎这个东西会不会出现在测验上）。这些都是些见风使舵的人。

11. 它老爸都自身难保了 .Sun 的商业模式正在从两个方面上被打压。用在桌面电脑■

原文未完，请参考

http://developer.51cto.com/art/201209/357654.htm

浅谈大型网站的算法和架构

上个月老大给我们讲解了 " 浅谈大型网站的算法和架构 ",获益匪浅。由于篇幅太多(光数据结构大概就有 20 多种),我也没有办法一下全部吸收,故我边理解,边分章节与大家分享。

序

上个月老大给我们讲解了 " 浅谈大型网站的算法和架构 ",获益匪浅。由于篇幅太多(光数据结构大概就有 20 多种),我也没有办法一下全部吸收,故我边理解,边分章节与大家分享。

这周我查阅资料,来理解各个数据结构和算法。

推荐几本个人感觉不错的书籍 :——我把电子书放到 http://download.csdn.net/user/rtxbc 这里了,需要下载 , 到这里进行下载。

《指针的艺术 . 蔡明志》——我只看了 C 语言这一篇。C 语言个人感觉比较难的也就是指针了。

《数据结构 使用 C 语言 [朱战立]》——严蔚敏的也不错,可就是里面的很多语法都是抽象语法,无法运行。我个人如果没有办法在终端运行,很难印象深刻。

《算法导论》

为了学习下载的电子书(截个图):

算法结构



C 语言



介绍

1984 年, Pascal 语义的发明者和结构化程序设计创始者,沃斯提出 “算法 + 数据结构 = 程序 “,从而获得当年的图灵奖。

现今技术日新月异,互联网技术在不断的发展中,从而也翻开了历史的新篇章 , 这时有人提出了 “算法 + 架构 = 互联网程序 “。

生活在这个时代的程序员来说,这又意味着什么呢?

从这篇文章开始,我将与各位浅谈大型网站的算法和架构,今天先了解一下基础知识,然后我们进行逐步过渡。

查找算法(单机)

1. 有个无序数组。■

原文未完,请参考

http://developer.51cto.com/art/201209/358529.htm